Atty Docket No. 166.0001 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of:

ROACH ET AL. Group Art Unit: 2172

Serial No.: 09/957,459

Filed: September 21, 2001 Examiner: B. To

For: AN AUTOMATIC REAL-TIME FILE MANAGEMENT METHOD AND APPARATUS

APPELLANT'S BRIEF ON APPEAL

Frederick N. Samuels, Esquire
Reg. 34,715
CAHN & SAMUELS, LLP
1100 17th Street, N.W., Suite 401
Washington, D.C. 20036
(202) 331-8777 (Telephone)
(202) 331-3838 (Facsimile)
Attorney for Appellants

TABLE OF CONTENTS

APPELLANTS' BRIEF ON APPEAL

1. Real Party In Interest

Integrity PC Innovations, Inc. is the real party in interest.

2. Related Appeals

None.

3. Status of Claims

(1) Claims 1-18 and 34-59 are currently pending in this application and stand rejected. Claims 1, 34 and 59 are objected to. Claims 1-16, 18, 34-49 and 51-59 are subject to this appeal.

4. Status of Amendments

No amendments has been filed subsequent to final rejection.

5. Summary of the Claimed Subject Matter

a. Claim 1

Claim 1 is directed to a method of archiving files performed by a computing device. The method includes detecting an instruction by an operating system to perform an operation on an operating file. Exemplary support may be found at page 9, ¶40, lines 1-3; page 13, ¶48, lines 1-4, Fig. 3 element 210; page 14, ¶ 50, lines 1-3, Fig. 4, element 305; and page 15, ¶51, lines 1-3, Fig. 5 element 405. The method further includes capturing the operating file temporally proximate to the operation being performed on the operating file, responsive to the detection of the instruction. Exemplary support may be found at page 9, ¶40, lines 3-8; pg. 13, ¶48, lines 3-4, Fig. 3 element 210; pg. 14, ¶50, lines 3-4, Fig. 4 element 310; pg. 15, ¶51, lines 3-4, Fig. 5 element 410.

b. Claim 34

Claim 34 is directed to a method of archiving files performed by a computing device. The method includes detecting an instruction by an operating system to perform an operation on an operating file. Exemplary support may be found at page 9, ¶40, lines 1-3; page 13, ¶48, lines 1-4, Fig. 3 element 210; page 14, ¶ 50, lines 1-3, Fig. 4, element 305; and page 15, ¶51, lines 1-3, Fig. 5 element 405. The method further includes creating an archive file from the operating file and storing the archive file in a temporary first storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction. Exemplary support may be found at pg. 10, ¶42, lines 1-4. The method still further includes searching the first temporary storage location for the archive file responsive to the occurrence of a first event. Exemplary support may be found at pg. 10, ¶43, lines 6-11; pg. 12, ¶45, Fig. 2A element 100. The method still further includes moving the archive file to a second storage location responsive to a second event, the second storage location being a permanent storage location. Exemplary support may be found at pg. 11, ¶43, lines 20-27, Fig. 2B element 125; pg. 12-13, ¶46.

c. Claim 54

Claim 54 is directed to still another embodiment of a method for archiving files in a computing device. The method includes detecting an instruction by an operating system to perform an operation on an operating file. Exemplary support may be found at page 9, ¶40, lines 1-3; page 13, ¶48, lines 1-4, Fig. 3 element 210; page 14, ¶ 50, lines 1-3, Fig. 4, element 305; and page 15, ¶51, lines 1-3, Fig. 5 element 405. The method further includes capturing the operating file just before or just after the

operation being performed on the operating file, responsive to the detection of the instruction. Exemplary support may be found at page 9, ¶40, lines 3-8; pg. 13, ¶48, lines 3-4, Fig. 3 element 210; pg. 14, ¶50, lines 3-4, Fig. 4 element 310; pg. 15, ¶51, lines 3-4, Fig. 5 element 410.

d. Claim 59

Claim 59 is directed to yet another embodiment of a method for archiving files performed by a computing device. The method includes detecting an instruction by an operating system to perform an operation on an operating file. Exemplary support may be found at page 9, ¶40, lines 1-3; page 13, ¶48, lines 1-4, Fig. 3 element 210; page 14, ¶ 50, lines 1-3, Fig. 4, element 305; and page 15, ¶51, lines 1-3, Fig. 5 element 405. The method further includes creating an archive file from the operating file and moving the archive file to a first storage device temporally proximate to the operation being performed on the operating file, responsive to detecting the instruction. Exemplary support may be found at pg. 10, ¶42 to pg. 11, ¶43, Fig. 2A element 100. The method still further includes storing the archive file in a second storage location. Exemplary support may be found at pg. 11, ¶43, Fig. 2B element 125.

6. Grounds of Rejection to be Reviewed on Appeal

(a) Whether claims 1-18 and 54-57 are unpatentable under 35 U.S.C. §103(a) over Koshisaka, U.S. Patent No. 6,629,109 B1 in view of Dunphy, U.S. Patent No. 5,638,509 and further in view of Parthasarathy, U.S. Patent No. 7,117,371 B1.

(b) Whether claims 34-38, 43-51 and 57-59 are unpatentable under 35 U.S.C. §103(a) over Dunphy, U.S. Patent No. 5,638,509 further in view of Koshisaka U.S.

Patent No. 6,629,109 B1 and further in view of Parthasarathy, U.S. Patent No. 7117,371 B1.

(c) Whether claims 39-42 are unpatentable under 35 U.S.C. §103(a) over U.S. Patent No. 5,638,509 in view of Koshisaka U.S. Patent No. 6,629,109 B1 and further in view of Parthasarathy, U.S. Patent No. 7117,371and further in view of Midgley, U.S. Patent No. 5,608,865.

7. Argument

a. Rejections (a) – (c)

A claimed invention is unpatentable for obviousness if the differences between it and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art. *In re Zurko,* 258 F.3d 1379, 1383 (Fed. Cir. 2001). Obviousness is a legal question based on underlying factual determinations including 1) the scope and content of the prior art, 2) the level of ordinary skill in the art, 3) the differences between the prior art and the claimed invention and 4) objective evidence of secondary considerations. Here the Official Action has erred in its factual assessment of points 1, 3 and 4 and, therefore, the rejections of all pending claims must be reversed. *Id at* 1386.

The Official Action made an incorrect factual finding that the claim term "operating system" was equivalent to the API of Koshisaka. The Official Action improperly relied upon this factual finding as a basis for its obviousness conclusions in all of the rejections. The Office Action fundamentally misconstrued the scope and meaning of the term API as used in Koshisaka. The record evidence, including

Koshisaka, Dunphy, The Webster's New World Dictionary of Computer Terms[1], compel the conclusion that Koshisaka's API is not equivalent to the claimed operating system.

Koshisaka itself clearly distinguishes between operating systems and application programs. See elements 1 and 3 of Figure 1. More particularly, Koshisaka defines Application (from which API commands emanate) as generally used application software such as word processor software, spreadsheet software, CAD software, etc. In contrast, Koshisaka defines operating system as software for operating the computer system, such as Windows 98.

Further in support of Applicants' position, one of the other cited references, Dunphy, discloses an operating system 19 that is separate and distinct from application programs 8. See Column 3, lines 36-47 and Figure 1.

Moreover, applicable technical dictionaries recognize the difference between APIs and operating systems. The Webster's New World Dictionary of Computer Terms defines "API" as a set of standards or conventions by which programs can call specific operating system or network services. The same dictionary defines the plain meaning of "Operating System" as a master control program that manages the computer's internal functions, such as accepting keyboard input, and that provides a means to control the computer's operations and file system. See pages 33 and 338 of Exhibit 1.

The Office Action alleges that Parasarathy discloses that its API is part of its operating system. For this assertion, the Office Action relies on Col. 5, lines 59-61. However, the cited passages of Parasarathy address neither operating systems nor APIs.

[1] Attached as Exhibit 1 to the Evidence Appendix of the Appeal Brief.

> ***"...hash is then encrypted using a private key that matches the public key located in the identity information component 17. The encrypted hash can be used to verify that the contents..."***

Col. 5, lines 59-61.

Nothing in this cited passage suggests to a person of ordinary skill in the art that an API is a part of the operating system.

Consequently, not a shred of evidence supports the Office Action's position that Koshisaka's API and the claimed operating system are the same. Because the Office Action's conclusion of obviousness in each rejection was based on this erroneous assessment of the scope of the prior art, all of the rejections lack substantial evidence support and, for this reason alone, must be reversed. *Zurko* 1386.

Even if Parasarathy taught what the Office Action alleges, the Office Action must view the prior art as a whole through the prism of a person having ordinary skill in the art. Koshisaka, Dunphy, the computer dictionary, Mr. Williams Declaration and the present specification teach that the operating system is separate from the API. Accordingly, the prior art taken as a whole does not support and, in fact, teaches away from the Office Action's conclusions the API is part of the operating system.

b. The Rejection of Claims 1-18 and 54-57 under 35 U.S.C. §103(a)

Turning now to the individual rejections, the Office Action rejected claims 1-18 and 54-57 under 35 U.S.C. §103 as unpatentable over Koshisaka in view of Dunphy and further in view of Parthasarathy et al., U.S. Patent No. 7,117,371 B1. This rejection is improper for the reasons set forth above as well as the following. The proposed combination of Koshisaka/Dunphy does not address all of the limitations of the claims.

For the purpose of this discussion, claim 1 is representative of claims 2-3, 9-12, 15 and 54-57. Claim 1 is directed to a method for archiving files and recites "detecting an instruction by an operating system to perform an operation on an operating file". Notwithstanding the comments to the contrary in the Office Action, neither Koshisaka nor Dunphy, taken alone or in combination, teach or disclose this detecting step.

As mentioned above, the Office Action errs in equating Koshisaka's API with the operating system of the claims. The term "operating system" is defined in the instant specification at paragraph 28 as:

> **"A computer program that allocates system resources such as memory, disk space, and processor usage and makes it possible for the computer to boot up to a human user interface allowing the user to interact with the computer and control its operation".**

Where an explicit definition is provided by the applicant for a term, that definition will control interpretation of the term as it is used in the claim. *Toro Co. v. White Consolidated Industries Inc.*, 199 F.3d 1295, 1301, 53 USPQ2d 1065, 1069 (Fed. Cir. 1999). Here, this axiom is particularly applicable as the definition of the term "operating system" set forth in the specification parallels the plain meaning of the term as evidenced by Exhibit 1 attached to the appeal brief and the understanding of the skilled artisan as evidenced by Koshisaka and Dunphy.

Koshisaka, teaches "an application-centric" file revision management system and method by which file revision management allegedly can be implemented even if applications are not provided with file revision management functions. Koshisaka teaches the use of an Applications Programming Interface (API) layer to be placed between an application and an operating system. According to Koshisaka, file backup reliability of the applications can be improved by this file revision management system.

See Koshisaka, column 1, lines 57-63. *See also* Exhibit 4, Paragraph 5. A file manipulation monitoring section in Koshisaka first detects file manipulation (e.g., file deletion or file name change) that is to be executed by the application by intercepting these instructions as they are generated by the application. *See* Exhibit 4, Paragraph 6. Koshisaka specifically states, "the file manipulation monitoring section constantly monitors API (Application Program Interface) commands which are outputted by the application 1 to the operating system and thereby detects the file manipulation which is (going to be) executed by the application 1." *See* Koshisaka, column 6, lines 34-38.

The method of the present invention, as defined by claims 1 and 54, is a "data-centric" approach to file archiving, not an application-centric approach. This distinction is evidenced by the claim limitations of, "detecting an instruction ***by an operating system*** to perform an operation on an operating file," and "capturing the operating file temporally proximate to the operation being performed on the operating file, responsive to the detection of the instruction."

In direct contrast, Koshisaka is not data-centric; that is, it does not teach detection of an instruction by an operating system. Rather, the detected ***instruction*** or command in Koshisaka is ***the API command*** that is generated ***by the application***, not ***the operating system***. *See* Exhibit 4, Paragraphs 5, 6 and 7. For example, in Koshisaka, when an API command requesting file deletion is output by the application, the command is detected and hooked by the file manipulation monitoring section in the file management system 2. Subsequently, the processing section sends a different API command to the operating system. In other words, the instruction is first detected by the API and hooked. After the instruction is hooked, a different instruction is passed to

the operating system, and the original command sent by the API to the operating system is not executed during performance of Koshisaka's revision management system activities. *See* Williams declaration, Paragraph 5.

As a result of detecting the instruction ***by the application***, unlike the present invention, it is believed that Koshisaka provides a very limited layer of file protection, as file protection occurs only if the application is compatible with Koshisaka's assumptions of API behavior. See, Koshisaka, column 7, lines 59-64. *See also* Exhibit 4, Paragraph 8. However, because the invention as defined by claim 1 is captures files based on operating system instructions, it achieves file protection of intended file change operations as well as file changes that result from some type of file corruption.

Dunphy is directed to a data storage and protection device used for data backup. Dunphy, like Koshisaka, teaches an application-centric solution. As illustrated in Figure 1, Dunphy depicts an operating system 19, application programs 8 and file system 9. Data file monitor 11 is interposed between application programs 8 and file system 9. Data file monitor 11 intercepts communications between ***application programs 8 and file system 9***. Data file monitor 11 reviews the communication to determine whether it relates to a data file that the user has selected for monitoring. If the data file is one to be monitored, it is determined whether the communication results in a change in content of a data file. If data file change is detected, data file monitor 11 extracts data file status and activity information from the received communications and uses this data to build an event log 12. Data file monitor 11 also determines whether the communication requests an operation that changes the contents of the data file, i.e., would cause a loss of data. If so, then the data file is saved. *See* Dunphy, Column

3 line 49 to column 4, line 21.

Unlike the method of claim 1, neither Dunphy nor Parasarathy teach detecting an instruction ***by an operating system***. The portion of Parasarathy that the Office Action relies on for such a teaching does not even mention APIs or operating systems.[2] Thus, Dunphy and Parasarathy suffer from the same deficiencies as Koshisaka.

Since the combination of Koshisaka, Dunphy and Parasarathy does not address all claim limitations, the rejection of claims 1-3, 9-12, 15 and 54-57 must be reversed.

c. The Rejection of claims 34-38, 43-51 and 57-59

The Office Action rejected claims 34-38, 43-51 and 57-59 under 35 U.S.C. §103 as unpatentable over Dunphy in view of Koshisaka and further in view of Parasarathy. This rejection is improper for the reasons set forth above as well as the following. The proposed combination of Dunphy, Koshisaka and Parasarathy does not address all of the limitations of the rejected claims.

Claim 34 is directed to a method for archiving files including the steps of detecting an instruction by an operating system to perform an operation on an operating file and creating an archive file from the operating file and storing the archive file in a temporary storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction. As mentioned above in the discussion of the rejection of claims 1-3, 9-12, 15 and 54-57, neither Dunphy, Parasarathy nor Koshisaka, taken alone or in combination teach or suggest detection of an instruction ***by an operating system***. In addition, neither Dunphy, Parasarathy nor Koshisaka teach storing an archive file created from the operating file in a temporary

[2] See Col. 5, lines 59-61 of Parasarathy.

first storage location responsive to detection of the operating system instruction. Koshisaka teaches that, upon detection of only one of a delete command or a file rename command from the ***application***, the deleted file name is stored and a corresponding back up file name is stored in memory. See Koshisaka, column 7, line 52 to column 9, line 43.

Unlike the method of claim 34, Dunphy **does not** detect an instruction ***by an operating system***. Dunphy, much like Koshisaka and other references of record, is concerned with communications from the application programs themselves. As explained in detail in connection with the discussion of Koshisaka above, the present invention as defined by claim 34 performs file capture and file manipulation based on instructions from the operating system. This is a significant advance because it allows file capture before the file operation is performed, for example. Dunphy is limited to addressing files for which an operation that changes the file content is specified, e.g., a delete or modify operation. Dunphy would be useless against operations that do not intentionally change file content but that may corrupt file content such as file open operations or file rename operations.

The cited passages of Parasarathy have no apparent relevance to the claims.[3]

It is readily apparent that the Office Action erred in its factual findings of the differences between the Dunphy/Koshisaka combination and the subject matter of claim 34. In view of this error, the Office Action failed to establish a *prima facie* case of obviousness as to claims 34-38, 43 and 59.

[3] See Col. 5, lines 59-61 of Parasrathy.

i. Claim 44

With respect to claim 44, it requires that the archive file pass through two storage locations before ending up in permanent storage (its third storage location). The Office Action cites to column 4, lines 25-67 of Dunphy as teaching the method of claim 44. However, the cited portion of Dunphy does not provide such a teaching.

Lines 24-38 of Dunphy discuss creation of an event log 12. Event log 12 is not an archive file. Rather it is a collection of data that includes identifying information about a file. The closest thing that Dunphy teaches to an archive file is the data file saved in stash can 13. However, Dunphy does not teach or suggest moving that data file from stash can 13 to an intermediate storage location and subsequently to a permanent storage location as required by claim 44. Koshisaka provides no additional teaching that would have rendered the subject matter of claim 44 obvious to the skilled artisan.

The Office Action's explanation of the operation of Dunphy on page 5 is unsupported by Dunphy itself. Database 14 and event log 12 are part of protection system 10. Accordingly, the explanation provided by the Office Action on page 5 is incorrect.

It is apparent that the Office Action erred in its factual findings of the differences between the Dunphy/Koshisaka/Parasarathy combination and the subject matter of claim 44. In view of this error, the Office Action failed to establish a *prima facie* case of obviousness as to claims 44-51 and 58.

d. The Rejection of Claims 39-42

The Office Action rejected claims 39-42 under 35 U.S.C. § 103 as unpatentable

over Dunphy in view of Koshisaka and further in view Parasarathy and further in view of Midgely et al., U.S. Patent No. 5,608,865 (hereinafter Midgely). This rejection is improper for the reasons set forth above as well as the following reasons.[4] The proposed combination of Dunphy, Koshisaka, Parasarathy and Midgely, does not teach all of the limitations of claims 39-42.

i. Claims 39 and 40

Claim 39 calls for searching a first storage location for the archive file responsive to receipt of a message from a timer. Accordingly, the storage location is searched at some specified time interval. The Office Action recognizes that neither Koshisaka nor Dunphy teach searching a storage location for the archive file responsive to a message from a timer. The Office Action asserts that Midgely suggests notifying a user that a file change is about to be made via a message from a timer. Office Action, Page 18. Applicants submit that the Office Action's assertion about Midgely's teachings are completely unsupported. Nowhere does Midgely even remotely indicate to a person having ordinary skill in the art that a temporary file location is searched responsive to a message from a timer as required by claim 39. The Office Action references a passage of Midgely, specifically column 7, lines 59-63, but this passage has nothing to do with the relevant claim limitations. At best, Midgely teaches generating a notification when a user requests a file open operation. It does not suggest the claimed operation of searching a temporary storage location responsive to any type of notification, whether that notification is a message from a timer as in claim 39 or a message from a resident program as in claim 40. Accordingly, the Office Actions failed to establish a *prima facie*

[4] The Office Action's reliance on Midgley is particularly troubling as Applicants

case of obviousness for claims 39 and 40 and the rejection of claims 39 and 40 must be reversed.

ii. Claims 41 and 42

Regarding claims 41, it calls for moving the archive file to a permanent storage location responsive to a message from a timer. Claim 42 calls for moving the archive file to a second storage location responsive to a message indicating when the second storage location is available. As recognized by the Office Action, neither Koshisaka nor Dunphy teach a method of archiving a file wherein archive files are moved to a second storage location responsive to receipt of a message. While the Office Action relied on Midgely for teaching that an agent is notified when a client requests a file open operation, prior to executing the open operation, Midgely offers no teaching that is remotely related to the limitations of claims 41 and 42. More particularly, Midgely does not teach moving an archive file responsive to a permanent storage location responsive to a timer message or a message indicating availability of the permanent storage location. Accordingly, the Office Action failed to establish a *prima facie* case of obviousness for claims 41 and 42 and the rejection of claims 41 and 42 must be reversed.

---

distinguished the claims from Midgley in an amendment filed on February 18, 2004.

Conclusion

In view of the foregoing arguments, it is respectfully submitted that claims 1-18 and 34-59 are allowable. Reversal of the rejections and allowance of all claims on appeal are respectfully requested.

Respectfully submitted,

CAHN & SAMUELS, LLP

By: __/Frederick Samuels/__
Frederick N. Samuels, Esq.
Reg. No. 34,715
11000 17th Street, N.W., Suite 401
Washington, D.C. 20036
(202) 331-8777 (Telephone)
(202) 331-3838 (Facsimile)
Attorney for Appellants

8. CLAIMS APPENDIX

1. In a computing device, a method for archiving files comprising:

detecting an instruction by an operating system to perform an operation on an operating file; and

capturing the operating file temporally proximate to the operation being performed on the operating file, responsive to the detection of the instruction.

2. The method of claim 1 wherein capturing the operating file includes creating an archive file and storing the archive file in a storage location.

3. The method of claim 2 wherein the archive file includes a copy of the operating file.

4. The method of claim 2 wherein the archive file includes portions of the operating file.

5. The method of claim 4 wherein the archive file includes pointers directed to one or more storage locations, wherein each of the one or more storage locations contains at least a portion of the operating file.

6. The method of claim 2 wherein capturing the file includes saving the archive file prior to the operation being performed on the operating file.

7. The method of claim 2 wherein capturing the file includes saving the archive file subsequent to detecting the instruction to perform the operation.

8. The method of claim 2 wherein capturing the file includes saving the archive file subsequent to the operation being performed on the operating file.

9. The method of claim 2 wherein the storage location includes a buffer.

10. The method of claim 2 wherein the storage location includes a storage device.

11. The method of claim 10 wherein the storage device includes at least one of a group comprising a magnetic storage medium, an optical storage medium, and a solid-state storage device.

12. The method of claim 10 wherein the storage location includes a directory disposed on said storage device.

13. The method of claim 1 further comprising determining whether the operating file is intended to be captured prior to said capturing step.

14. The method of claim 1 further comprising determining whether the operating file has previously been captured prior to capturing the file.

15. The method of claim 1 further comprising determining whether the operation causes a change in the operating file.

16. An article of manufacture comprising a computer usable medium having computer readable program code for performing the method of claim 1.

18. An article of manufacture comprising a processor configured to perform the method of claim 1.

34. In a computing device, a method for archiving files comprising:

detecting an instruction by an operating system to perform an operation on an operating file;

creating an archive file from the operating file and storing the archive file in a temporary first storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction;

searching the first temporary storage location for the archive file responsive to the occurrence of a first event; and

moving the archive file to a second storage location responsive to a second event, the second storage location being a permanent storage location.

35. The method of claim 34 wherein storing the archive file includes storing the archive file prior to the operation being performed on the operating file.

36. The method of claim 34 wherein storing the archive file includes storing the archive file prior to the operation being performed on the operating file and subsequent to the operation being performed on the operating file.

37. The method of claim 34 wherein storing the archive file includes storing the archive file subsequent to the operation being performed on the operating file.

38. The method of claim 34 wherein the first temporary storage location includes a buffer.

39. The method of claim 34 wherein the first event includes a message from a timer.

40. The method of claim 34 wherein the first event includes a message from a program resident on the computing device.

41. The method of claim 34 wherein the second event includes a message from a timer.

42. The method of claim 34 wherein the second event includes a message indicating when the second storage location is available.

43. The method of claim 34 wherein the second storage location is an output buffer.

44. The method of claim 34 further comprising:

after storing the archive file in the first temporary storage location, updating a database to indicate that the archive file is located in the first temporary storage location;

determining a final destination for the archive file;

moving the archive file from the first temporary storage location to an intermediate storage location;

updating the database to indicate that the archive file is located in the intermediate storage location; and

after moving the archive file to the second storage location, updating the database to indicate that the archive file is located in the second storage location.

45. The method of claim 44 wherein the second storage location includes a personal attached storage device.

46. The method of claim 44 wherein the second storage location includes a network attached storage device.

47. The method of claim 44 wherein the second storage location includes a peer-to-peer storage device.

48. The method of claim 44 wherein the second storage location includes an

Internet storage area network.

49. An article of manufacture comprising a computer usable medium having computer readable program code for performing the method of claim 44.

51. An article of manufacture comprising a processor configured to perform the method of claim 44.

52. The method of claim 2, wherein said capturing step occurs only if a match to a defined condition has been determined.

53. The method of claim 52, wherein said defined condition includes at least one of determining whether the operating file has previously been archived and determining whether the operating file has been selected for protection.

54. In a computing device, a method for archiving files, comprising:

detecting an instruction by an operating system to perform an operation on an operating file; and

capturing the operating file just before or just after the operation being performed on the operating file, responsive to the detection of the instruction.

55. The method of claim 54, wherein said capturing occurs an instant before or an instant after the operation is performed on the operating file.

56. The method of claim 54, wherein the operating file is a system file.

57. The method of claim 54, wherein the operating file is a user file.

58. The method of claim 34, wherein said first event is different from said

second event.

59. In a computing device, a method for archiving files comprising:

detecting an instruction by an operating system to perform an operation on an operating file;

creating an archive file from the operating file and moving the archive file to a first storage device temporally proximate to the operation being performed on the operating file, responsive to detecting the instruction; and

storing the archive file in a second storage device.

10. EVIDENCE APPENDIX

11. RELATED PROCEEDINGS APPENDIX

NONE

Exhibit 1

Exhibit 1

WEBSTER'S NEW WORLD™

DICTIONARY of COMPUTER TERMS

EIGHTH EDITION

The Best Computer Dictionary in Print

Completely revised and updated

Contains extensive Internet coverage

More than 4,000 entries, terms, and acronyms

Bryan Pfaffenberger

THE NAME YOU TRUST

Dedication

For Suzanne, always

Webster's New World™ Dictionary of Computer Terms,
8th Edition

For general information on IDG Books Worldwide's books in the U.S., please call our Consumer Customer Service department at 1-800-762-2974. For reseller information, including discounts, bulk sales, customized editions, and premium sales, please call our Reseller Customer Service department at 1-800-434-3422.

A Webster's New World™ Book

Library of Congress Catalog Number: 98-68180

ISBN: 0-02-863777-1

Manufactured in the United States of America

1 2 3 4 5 6 7 00 1 02 03 04

share

: A file server utility for AppleTalk networks. transforms any Macintosh on the network into file server; the server's hard disk icon appears on ork user's desktop.

A small- to medium-sized computer program that specific function, such as emulating a calculator. **2.** In program embedded in a Web document that, when d, is executed by the browser. Both of the leading Netscape Communicator and Microsoft Internet can execute Java applets. See *Java applet* and *Java*

lk A local area network (LAN) standard developed by omputer. AppleTalk can link as many as 32 Macintosh rs, IBM PC–compatible computers, and peripherals aser printers. Every Macintosh computer has an lk port; the only hardware required for an AppleTalk is a set of LocalTalk connectors and ordinary telephon cables (called twisted-pair cable). AppleTalk networks ple and inexpensive but quite slow; they are capable of tting only up to 230 Kbps. EtherTalk, in contrast, is of speeds of up to 10 million bps. See *EtherTalk.*

ation A program that enables you to do something with the computer, such as writing or accounting posed to utilities, programs that help you maintain omputer).

ication Configuration Access Protocol (ACAP) osed Internet standard that transfers crucial user confi settings (including address books, bookmarks, and op ces) to an Internet-accessible file. Because these settin d on the network instead of the user's computer, the ssible no matter which computer is being used. ACA tly benefit anyone who accesses the Internet from m computer.

plication control menu See *control menu.*

plication development system A coordinated am development tools, typically including a full-scr programming language with a compiler, linker, and

and an extensive library of ready-to-use program modules. The use of an application development system lets experienced users develop a standalone application more easily than writing a program using a language such as C++ or COBOL.

application heap In a Macintosh, the base memory, the area of memory set aside for user programs.

application icon In Microsoft Windows 95/98, an onscreen graphic representation of a minimized program. The icon appears on the taskbar to remind you that the application is still present in memory. Double-click the application icon to switch to that program.

application layer In the Open System Interconnection (OSI) reference model of computer network architecture, the first or highest of seven layers, in which the data is presented to the user. At this layer, protocols are needed to ensure that products made by different manufacturers can work together. For example, every e-mail program should use the same protocols for sending and receiving e-mail. When the data is ready to be sent on the network, it is passed down the protocol stack to the next layer, the presentation layer.

application-level encryption In a computer network, the implementation of encryption by individual applications rather than at the operating system or network level. Web browsers typically implement encryption at this level.

application program See *application.*

application program interface (API) **1.** A set of standards or conventions by which programs can call specific operating system or network services. **2.** In Web servers, the standards or conventions that enable a hyperlink to originate a call to a program that executes on the server See *CGI, ISAPI,* and *NSAPI.*

application shortcut key In Microsoft Windows, a shortcut key combination (hot key) for bringing an application to the foreground. Application shortcut keys are also available in applications such as DESQview and PC Tools Desktop to launch and switch among programs.

operand The argument that is appended to an operator, such as a spreadsheet program's built-in function. For example, in the Excel expression AVERAGE(D10:D24), the cell range D10 to D24 is the operand of the AVERAGE function.

operating environment The total context in which applications function, including the operating system (OS) and the shell.

operating system (OS) A master control program that manages the computer's internal functions, such as accepting keyboard input, and that provides a means to control the computer's operations and file system.

operating voltage The electrical voltage at which a microprocessor operates. Most microprocessors have operating voltages of 5 volts—a mostly arbitrary specification decided upon when the transistor was invented—but some chips run at 3.3 volts to save electricity (a real concern in portable computers) and reduce heat output.

operator In programming, a code name or symbol that is used to describe a command or function, such as multiplying or dividing.

optical character recognition See *OCR*.

optical disk A large-capacity data storage medium for computers on which information is stored at extremely high density in the form of tiny pits. The presence or absence of pits is read by a tightly focused laser beam. CD-ROMs and CD-R drives offer an increasingly economical medium for read-only data and programs. Write-once, read-many (WORM) drives enable organizations to create their own huge, in-house libraries. Erasable optical disk drives offer more storage than floppies and the CDs are removable. However, they are still relatively expensive and much slower than hard disks. See *interactive*.

optical fiber See *fiber optics*.

optical mouse A mouse that does not require a ball like a mechanical mouse does, but that must be used on a special mouse pad. An optical mouse shines a beam of light on a grid in the mouse pad, which conveys the mouse's movements to the computer.

Exhibit 2

Exhibit 2

Atty. Docket No. 166.0001

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : Roach *et al.*

Serial No. : 09/957,459 Examiner: To, Baoquoc N.

Filed : September 21, 2001 Group Art Unit: 2172

For : AN AUTOMATIC REAL-TIME FILE MANAGEMENT METHOD AND APPARATUS

DECLARATION UNDER 37 CFR §1.132

I, Steven R. Williams, declare under penalty of perjury, that:

1. I am an electrical engineer and Vice President of precisionWave Corporation with approximately 23 years of experience in the software development industry. I make this declaration based upon my knowledge and years of experience.

2. I earned a Bachelor of Science degree in electrical engineering from the University of Illinois in 1981.

3. I am listed as a co-inventor on U.S. Patent Application No. 09/957,459.

4. I have reviewed U.S. Patent No. 6,629,109 B1, issued to Koshisaka (hereinafter "Koshisaka").

5. Koshisaka teaches "an application-centric" file revision management system for executing file revision management when an application operating on an operating system of a computer system saves a file by file overwrite. The file manipulation monitoring section monitors and hooks Application Program Interface (API) commands outputted by the application to the operating system, thereby detecting file manipulation executed by the application. After a command is hooked, a different instruction is passed to the operating system in order to carry out Koshisaka's revision management system activities. *See* Koshisaka, column 6, lines 32-43.

6. As illustrated by EXHIBIT A, a diagram of a likely implementation of the Koshisaka file revision management system, the system of Koshisaka includes an application 105 that is written for the Koshisaka API, a Koshisaka-specific API 110, an

operating system 115, and a group of user files 120. As Koshisaka is an application-centric system, the file manipulation monitoring section of Koshisaka detects an instruction 125 (for example, a file deletion instruction) which is going to be outputted *by the application*. Koshisaka does not operate by detecting an instruction by an operating system. *See* Koshisaka, column 6, lines 32-43. Thus, Koshisaka fails to teach or suggest at least the "detecting an instruction *by an operating system*" limitation of the claims. Therefore, it would not have been obvious to one of ordinary skill in the art at the time the above-referenced invention was made to modify the teachings of Koshisaka to obtain the above-referenced invention.

7. An alternative implementation of Koshisaka includes an application that is NOT written for a Koshisaka-specific API, but is written to output instructions directly to an operating system via the normal Windows API. Koshisaka suggests the possibility of monitoring such outputted instructions from the application to the operating system, and supplanting these operating system instructions with different operating system instructions in order to carry out Koshisaka's revision management activities. Even in this alternative implementation, Koshisaka cannot be interpreted to teach or suggest at least the "detecting an instruction *by an operating system*" limitation of the claims. As in item 6 above, it would not have been obvious to one of ordinary skill in the art at the time the above-referenced invention was made to modify the teachings of Koshisaka to obtain the above-referenced invention.

8. As a result of detecting the instruction by the application, unlike the invention disclosed in the above-referenced application, Koshisaka provides a very limited layer of file protection.

9. I have reviewed U.S. Patent No. 6,535,894, issued to Schmidt *et al.* (hereinafter "Schmidt").

10. Schmidt discloses an apparatus and method for incremental updating of archive files. An original archive file having one or more entries is created, where each entry in the original archive file is itself a file. The original archive file is transmitted to a client computer, and subsequently, a target archive file is created wherein one or more of its entries are "typically expected" to be identical to one or more entries in the original archive file. A difference file including an index file describing the changes between the

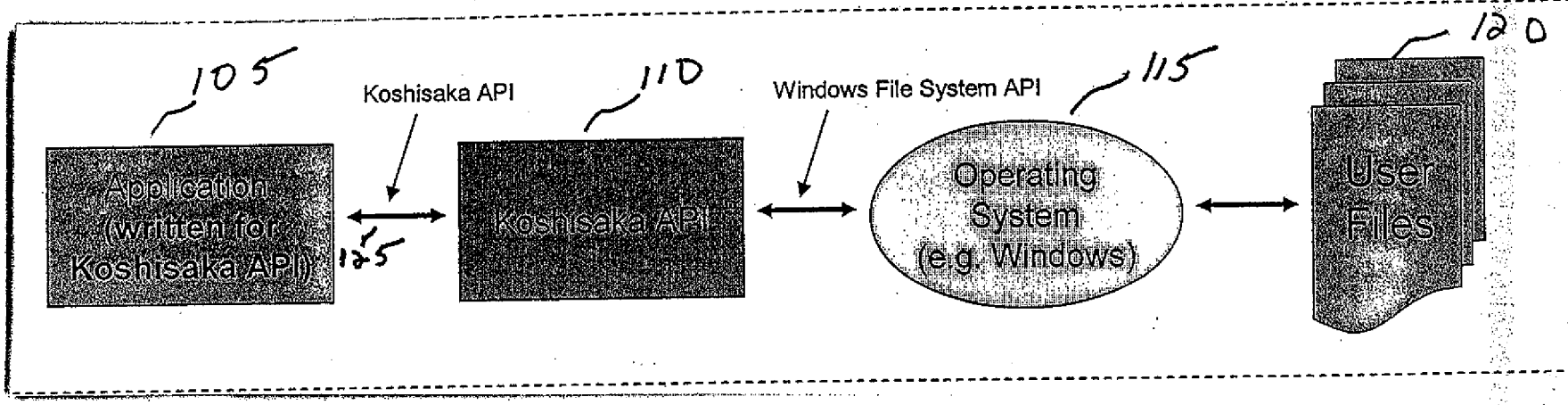
original archive file and the target archive file is also created and transmitted to the client computer. At the client computer, Schmidt teaches that the difference archive file is applied to the original archive file to produce a synthesized archive file. *See* Schmidt, column 9, line 65 – column 10, line 59.

11. Schmidt's field of invention relates to an apparatus and method to facilitate incremental updating of program code. *See* Schmidt, column 1, lines 8-14. Schmidt discloses methods for optimizing transmitted archive files through the creation and manipulation of original archive files, target archive files, difference archive files and synthesized archive files. Schmidt teaches improvements to the deployment of program code from server computers to client computers.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Executed this 7th day of September 2004.

Steven R. Williams


105
Application (written for Koshisaka API)
Koshisaka API
125
110
Koshisaka API
Windows File System API
115
Operating System (e.g. Windows)
User Files
120